Seppo Luong

Reseptinhallintaohjelma

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Automaatiotekniikka
Insinöörityö
25.4.2013

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Seppo Luong Reseptinhallinta ohjelma 22 sivua + 4 liitettä 25.4.2013
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	Kappaletavara-automaatio
Ohjaaja(t)	Manager Prosess Development Harri Hyppönen Lehtori Timo Tuominen

Insinöörityössä oli tavoitteena parantaa laitteiden prosessikehityksen ja huollon tiedonkulkua. Työ toteutettiin Murata Electronics Oy:n komponenttivalmistuksen kokoonpanoon. Vastaavanlainen ohjelma oli jo käytössä Murata Electronicsin komponenttivalmistuksen testi puolella ja se oli todettu toimivaksi.

Muratan komponenttivalmistuksen kokoonpano on toiminnassa vuoden jokaisena päivänä ympäri vuorokauden, joten käyttäjiä on paljon. Laitteisiin tehdyt muutokset ja huollot eivät aina kulkeudu jokaiselle käyttäjälle, mikä tuotti ongelmia käyttäjille.

Työssä kartoitettiin erilaisia versionhallintaohjelmia. Kartoituksen ja palaverien jälkeen päätettiin kokeilla ottaa käyttöön Tortoise SVN -versionhallintaohjelma. Tarkoituksena oli ottaa ohjelma koko kokoonpanoon, mutta ensin otetaan käyttöön Muratan Cobra-linjalla, joka valmistaa yhdistelmäantureita. Pilottilaitteeksi valitsimme Esec 3100 plus wire bonder -latitteen.

Pilottilaitteen valittua perehdyttiin ohjelmaan ja sen soveltuvuuteen tiedonkulkuongelmaan. Ohjelmaan lisättiin koodia, jotta se soveltuisi kokoonpanon käyttöön paremmin. Ottaessa ohjelmaa käyttöön pilottilaitteelle suurin osa ongelmista tuli Muratan verkkojen välisistä ongelmista. Niitä yritettiin ratkaista erilaisilla verkko-ohjelmilla. Aikataulun takia työ jää pohdintaan, jatketaanko sitä vai kokeillaanko jotain muuta ratkaisua.

Avainsanat	yhdistelmäanturi, Tortoise SVN, MEMS, versionhallinta

Metropolia

Abstract

Author(s) Title Number of Pages Date	Seppo Luong Recipecontrol system 22 pages + 4 appendices 25 April 2013
Degree	Bachelor of Engineering
Degree Programme	Automation Technology
Specialisation option	Manufacturing Automation
Instructor(s)	Harri Hyppönen, Manager, Process development Timo Tuominen Lecturer

This engineering thesis project was conducted to improve information flow of process development and maintenance in the Murata Electronics Oy component manufacturing assembly. At the Murata Electronics component manufacturing test side there already was already a similar program in use, which solved assembly's problem.

The Murata component manufacturing assembly side operates all the time, which means there are many users manufacturing devices. When there is maintenance or process development in the devices and the changed information does not reach each user, it creates many problems.

The first thing done was to study different types of version control softwares. After the mapping and the meetings we decided to use Tortoise SVN version control software. The purpose of the project was to take the software to the whole assembly but we decided to start with the Cobra line which produced gyroscopes. As the pilot device we chose the Esec 3100 wire bonder plus.

After the pilot device was selected, we started to study the Tortoise SVN version control software. I added some code into the program so that it fits better to use in our project. Most Most of the problems were between Murata's networks. We tried to solve the problem with different network programs. Because of the schedule, the work remains unfinished. It remains to be seen, whether the work will continue or, if it will have to be changed to try another solution.

Keywords	version control, MEMS, Tortoise SVN, gyroscopes

Sisällys

Merkinnät ja lyhenteet

ABS	Anti-lock braking system, lukkiutumaton jarrujärjestelmä
ARC	Anti roll control, kallistuksen esto
ASIC	Anturijärjestelmään sovitettu elektroninen integroitu piiri
Coriolis-voima	Maapallon pyörimisliikkeestä johtuva voima
CVS	Concurrent version system, versionhallintaohjelma
ECS	Electronic stability control, elektroninen jousitusjärjestelmä
Elektrodi	Sähköisen virtapiirin osa, jossa sähkö siirtyy väliaineeseen
EPB	Electronic parking brake, elektroninen seisontajarru
ESC	Electronic stability control, ajonvakautusjärjestelmä
ESD	Staattisen sähkövarauksen purkautumisilmiö
FTP	File transfer protocol, tiedonsiirtomenetelmä
HSA	Hill start assistance, mäkilähtöavustin
Kapasitanssi	Sähköstaattinen suure
Kinematiikka	Tutkii kappaleiden liikettä geometriseltä kannalta
MEMS	Mikro-elektro-mekaaninen systeemi
Novell	Verkkojärjestelmiin erikoistunut yhtiö
RCS	Reversion control system, versionhallintaohjelma
Resepti	Tiedosto, joka sisältää tuotteiden parametrit
ROV	Roll over, kaatumissuoja
SCCS	Source code control system, versionhallintaohjelma
Tortoise SVN	Versionhallintaohjelma
Unix	Laitteistoriippumaton käyttöjärjestelmä
Yhdistelmäanturi	Anturi kiihtyvyyden ja kulmanopeuden mittaamiseen

1 Toimeksiantaja

Työn toimeksiantaja on Murata Eelctronics Oy, joka sijaitsee Vantaan Martinlaaksossa osoitteessa Myllykivenkuja 6. Yrityksen toimialana on teknillisten kojeiden ja laitteiden kehitys, suunnittelu, valmistus ja markkinointi. Murata suunnittelee ja valmistaa MEMS-ratkaisuja kuljetusteollisuuteen, kulutuselektroniikkaan ja asiakaskohtaisiin tuotteisiin, esimerkiksi terveysteknologiaan. Yrityksen tärkeimpiä tuotteita ovat kiihtyvyys-, kallistus-, kulmanopeus- ja yhdistelmäanturit sekä anturielementit. Sovellusalueita ovat autoteollisuus, kulutuselektroniikka, terveysteknologia, instrumentit, raskaat ajoneuvot ja ilmailuteollisuus. Murata on globaali markkinajohtaja pienkiihtyvyysantureissa ja sydämentahdistimen liikeanturissa. Yrityksen liikevaihto on 75,8 miljoonaa euroa ja henkilöstöä on yli 600. Yrityksellä on lisäksi vahva asema rakennus- ja ilmailuteollisuuden instrumenttisovelluksissa. Murata Electronics Oy:n pääkonttori, tuotekehitys ja yhtiön oma MEMS -valmistus sijaitsevat Vantaan Martinlaaksossa. [10]

VTI perustettiin vuonna 1991, mutta teknologian kehittäminen alkoi Vaisalassa jo 1979. Vuonna 1993 yritys oli johtava pienkiihtyvyysantureiden toimittaja autojen jousitusjärjestelmässä. Seuraavana vuonna VTI oli maailman johtava pienkiihtyvyysantureiden toimittaja autojen jarru- ja luistonestojärjestelmissä. Yritys alkoi lisäksi toimittaa kiihtyvyysantureita sydämentahdistimiin. BREED Technologies osti VTI:n vuonna 1995. Vuosina 1998 ja 1999 uusi MEMS-tehdas avattiin Martinlaaksossa ja yritys oli maailman johtava pienkiihtyvyysantureiden toimittaja autojen ajovakauden hallintajärjestelmissä. Vuonna 2002 VTI siirtyi EQT III -pääomasijoitusrahaston omistukseen. Yritys toi markkinoille maailman pienimmän ja vähävirtaisimman 3-akselisen kiihtyvyysanturin vuonna 2009. Murata Manufacturing osti VTI:n vuonna 2012. [10]

2 Lähtötilanne

Murata Electonicsin komponenttivalmistuksen kokoonpanon Cobra-linjan valmistaa kulmanopeusantureita ja kallistusantureita. Yhdistelmäanturit mittaavat kiihtyvyyttä ja kulmanopeutta, kun taas kulmanopeusanturit mittaavat kulmanopeutta. Yhdistelmäanturit koostuvat kahdesta elementistä ja kahdesta asicista. Komponenttivalmistuksen kokoonpanolinja määrittää yhdistelmäanturit kolmeen tuotteeseen. Laitteelle on opetettu jokaiselle eri tuotteelle tietyt parametrit, ja niiden tiedot ovat tallennettu resepteihin. Näihin resepteihin haluttiin parempi kontrolli, kun reseptien muutosten tieto ei saavuttanut jokaista laitteen käyttäjää. Laitteista saattoi löytyä melkein samannimisiä tiedostoja, eikä tiedetty, kumpi tiedostoista oli uusin toimiva resepti. Ongelmana oli, että reseptinhallintaa yritettiin pitää pelkällä lokilla, mutta kun joitakin reseptejä ajetaan vähän harvemmin, käyttäjiä on useita eikä kommunikaatio ollut täydellistä, niin tieto ei aina mennyt jokaiselle laitteen käyttäjälle. Tiedettiin, että komponentti valmistuksen testipuolella on käytössä Tortoise SVN -versionhallintaohjelma reseptinhallintaan. Myös kokoonpanon Cobra-linjalla haluttiin ottaa kyseinen ohjelma käyttöön, mutta myös haluttiin tietää onko olemassa toinen versionhallintaohjelma, joka sopisi paremmin kokoonpanolinjalle. Kun kartoitus oli tehty, päätettiin ensin yrittää ottamaan käyttöön Tortoise SVN -ohjelmaa. Jos ohjelmaa ei saada toimimaan suurimmalla osalla linjan laitteista, aletaan mietitään jotain toista ratkaisua.

Aluksi piti selvittää, mihin Muratan komponenttivalmistuksen kokoonpanon Cobra-linjan laitteista saadaan versionhallinta toimimaan ilman minkäänlaista muutosta, mihin laiteeseen saadaan versionhallinta toimimaan pienillä muutoksilla ja mistä tiedetään, mitä pitää tehdä ja mihin laitteisiin ei saada toimimaan ilman suuria muutoksia. Projektin tehtävänä on ottaa käyttöön versionhallintaohjelma hallitsemaan Muratan kokoonpanon reseptinhallintaa yhdelle laitteelle. Ohjelman tehtävänä on reseptinhallinnan lisäksi nähdä uusimmat päivitykset, muutoksen syy ja se kuka sen on tehnyt sekä milloin muutos on tehty. Päätimme ottaa laiteeksi Esec:n 3100 Plus wire bonder -laitteen, koska Muratalla on yksi edellä mainittu wire bonder -laite, joka ei ole tuotannon käytössä. Täten ohjelmaa testatessa laitteella, ei hidastettaisi tuotantoa. Esec 3100 Plus wire bonder -laite yhdistää asicit, elementit ja kotelon toisiinsa kultalangalla.

3 Versionhallinta

Versionhallintaohjelma voidaan jakaa toimenpiteisiin versiointi, versioiden merkitseminen, versioiden välisen erojen tunnistaminen ja versioiden tallentaminen. Versioinnissa tehdään tiedostosta, hakemistosta tai oliotietokannan oliosta alkio, jota voidaan kehittää. Version merkitsemisessä reversiot identifioidaan yleensä numeraalisesti järjestyksen mukaan. Versioiden välien erojen tunnistamiseen yleensä käytetään merkkivertailua tekstitiedostojen välillä. Versioiden välisten erojen tunnistuksen avulla pystytään määrittämään eroavaisuudet ja voidaan keskittyä muutokseen sekä säästää resursseja, koska eri versiot saattavat olla toistensa kaltaisia. Ohjelmistojen kehityksen ja ohjelmiston aikaisempien versioiden uudelleenrakentamisen mahdollistamiseksi on vanhat ja uudet versiot voitava tallentaa pysyvästi. [1]

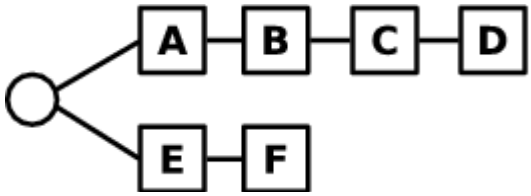
Hyvin yleinen tapa on käyttää tietovarastoa, jonne kaikki versiohallintatiedostot tallennetaan. Tietovarastoon voi tallentaa melkein mitä tahansa, kuten yksittäisen tiedoston, hakemiston tai tietokannan. Tietovarasto voi olla keskitetty, hajautettu, paikallinen tai se voi sijaita jossain muualla. Sen tehtävä on eri versioiden säilytys ja niihin pääsyn mahdollistaminen, ja ylläpitää muutoksiin liittyviä muutostietoja, esimerkiksi muutosten ajankohdat ja tekijät. Paikallisissa tietovarastoissa ovat tiedot aina saatavilla, eivätkä ne tarvitse verkkoyhteyttä. Lisäksi tietovaraston pääsynvalvontaa on helpompi toteuttaa, mutta muutosten jakelu tarvitsee enemmän työtä. Ulkopuolella tehdyt muutokset pitää ottaa mukaan paikallisesti ja tietovaraston varmuuskopioinnin voi joutua käyttäjä tekemään.

Keskitetyt tietovarastot sijaitsevat melkein aina verkossa, missä tietojen varmuuskopiointi on helppoa ja tietovarastoon pääsy voidaan estää tai sallia käyttäjäkohtaisesti. Siinä voidaan myös lähettää automaattinen ilmoitus kaikille asianosasille, kun muutos on tehty ja tallennuksen yhteydessä voidaan asettaa tarkistuksia muutoksille. Keskitetty tietovarasto vaatii kuitenkin asianmukaisen ylläpidon, jotta se toimisi sujuvasti. Hajautetussa tietovarastossa jokainen alkuperäisestä tietovarastosta tehty kopio on itsenäinen ja tasavertainen. Toisin sanoen

tietoja voidaan siirtää eri kopioiden välillä. Hyvänä puolena verkkoyhteyttä voidaan hyödyntää, mutta se ei ole pakollinen. Muutosten jakelu on helppoa, muttta pääsynvalvonta pitää konfiguroida tilannekohtaisesti. [1;2]

3.1 Versio, reversio ja variantti

Versio on kehittyvän alkion tila tiettynä hetkenä. Alkio voi olla mikä tahansa versionhallintaan laitettava, asia esimerkiksi tiedosto, kansio, hakemisto tai oliotietokannan olio. Reversio on alkion sellainen versio, joka korvaa edeltäjänsä. Variantti on alkion versio, jonka tarkoitus on olla olemassa samanaikaisesti muiden versioiden kanssa. Kuvassa 1 on esimerkki, jossa A, B, C, D, E ja F ovat alkion eri versioita. A, B, C ja D ovat toistensa reaversioita. E ja F ovat A, B, C ja D:n variantteja ja toistensa variantteja. Useimmat versionhallintaohjelmat tukevat tämänlaisia kehityshaaran käyttöä, mutta niiden toteutustapa vaihtelee. Kehityshaaran etuna on alkuperäisen version samanaikainen kehitys erillisissä haaroissa. [3]

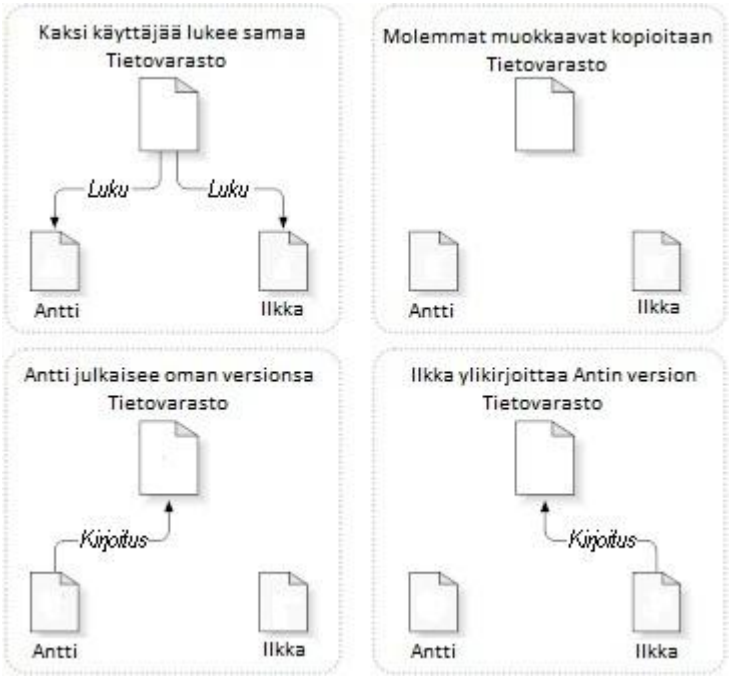


Kuva 1. Versio, reversio ja variantti [3]

3.2 Jaetun tiedoston ongelma

Se, miten ohjelma pystyy jakamaan tietoa, mutta ei anna käyttäjän vahingossa ylikirjoittaa toistensa muutoksia tietovarastossa, on jokaisen vesionhallintaohjelman päätehtävä. Hyvänä esimerkkinä Antti ja Ilkka tekevät projektia ja haluavat muokata samaa tiedostoa. Jos Antti tallentaa oman muutoksensa tietovarastoon ensin, niin Ilkalla on myöhemmin mahdollisuus vahingossa ylikirjoittaa Antin muutokset huomaamatta sitä, että Antti on tehnyt muutoksia alkuperäiseen tietovarastoon. On mahdollista, että Antin muutoksia ei näy enää uusimmissa versioissa ollenkaan, kun

Ilkka ei ole nähnyt Antin muutoksia. Tämänlaisilta tilanteilta pyritään välttymään versionhallintaohjelmalla. [3]

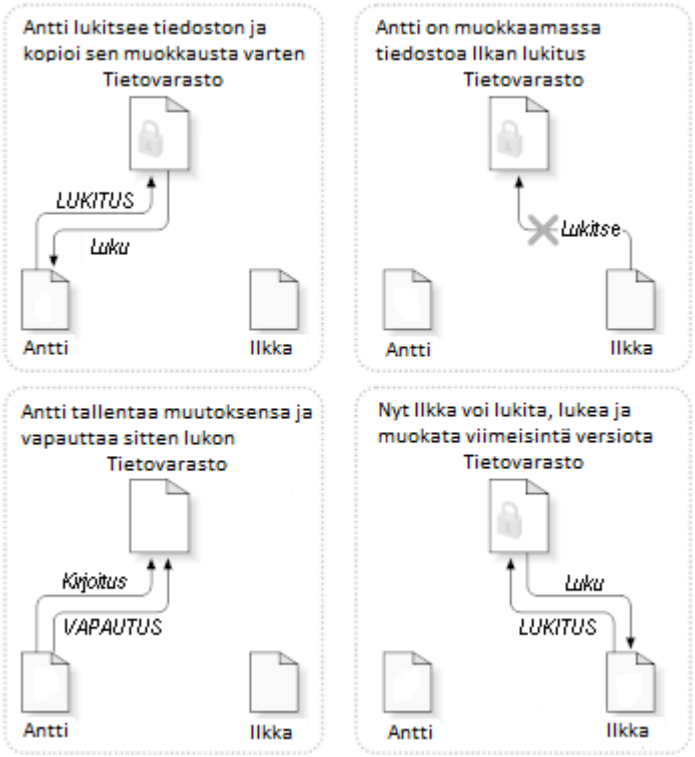


Kuva 2. Vältettävä ongelma. [3, muokattu]

3.3 Lukitse—muokkaa—vapauta-ratkaisumalli

Vanhat versionhallintaohjelmat käyttävät lukitse—muokkaa—vapauta-ratkaisumallia ratkaistakseen monen käyttäjän ylikirjaamisen ongelman. Tällaisessa mallissa tietovarasto antaa vain yhdelle henkilölle kerrallaan oikeuden kirjoittaa tietoa tietovarastoon. Tämä toimii niin, että Antin on ensin lukittava tietovarasto, jotta hän voi muokata tietovarastoon tallennettua tiedostoa. Ilkka ei pysty samaan aikaan lukitsemaan sitä samaa tiedostoa, jolloin Ilkka ei voi myöskään muokata kyseistä tiedostoa. Ilkka ainoastaan pystyy lukemaan tiedostoa ja odottamaan, että Antti on saanut muutokset tehtyä ja vapauttaa lukituksen. Vasta, kun Antti on vapauttanut lukituksen, Ilkka voi lukita ja muokata tiedostoa. Lukitse—muokkaa—vapauta-ratkaisumallin ongelmana on se, että jos Antti lukitsee tiedoston ja unohtaa vapauttaa sen ennen lomaa tai matkaa, niin silloin Ilkka joutuu odottamaan, että Antti vapauttaa tiedoston, tai Ilkka joutuu ottamaan järjestelmävalvojaan yhteyttä, joka avaa lukituksen. Tämän jälkeen Ilkka voisi muokata tiedostoa. Tässä ratkaisumallissa haittana on myös se, jos Antti haluaa muokata esimerkiksi tekstitiedoston alkuosaa ja

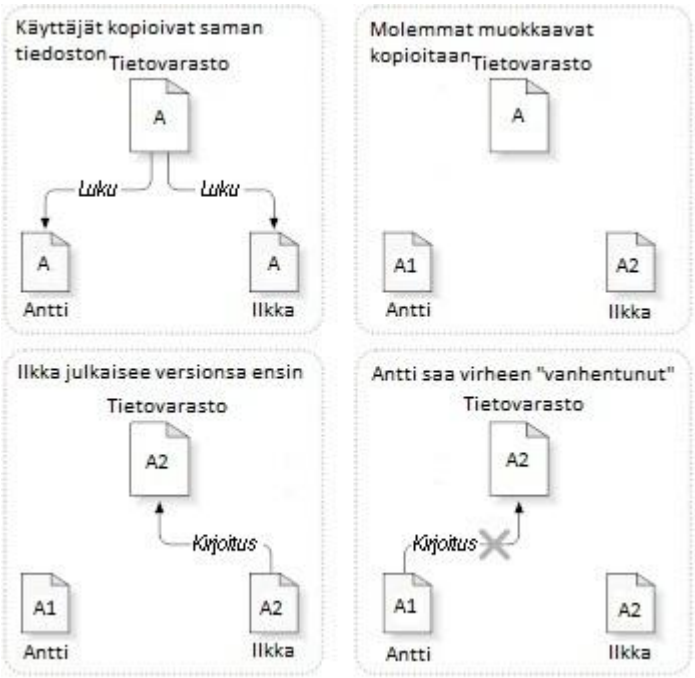
Ilkka loppuosaa. He voisivat samaan aikaan muokata samaa tiedostoa. Ei tarvitsisi odottaa vuoroaan, koska ne eivät ole ristiriidassa toistensa muutoksien kanssa. Tällöin tilanteiden ratkaisuun menee turhaa aikaa. [3]



Kuva 3. Lukitse—muokkaa—vapauta-ratkaisumalli. [3, muokattu]

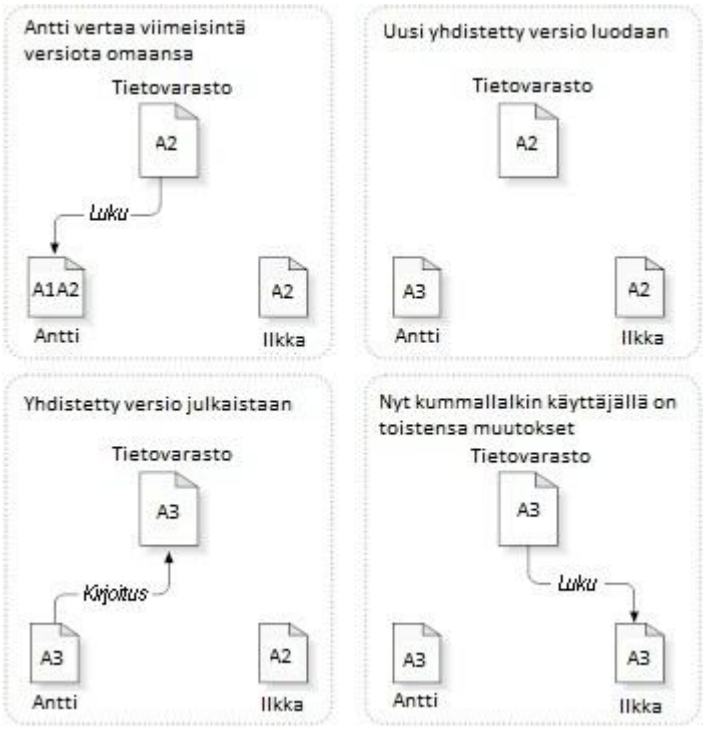
3.4 Kopioi—muokkaa—yhdistä-ratkaisumalli

Subversion, CVS ja monet nykyiset versionhallintajärjestelmät käyttävät kopioi—muokkaa—yhdistä-ratkaisumallia ja vaihtoehtona lukitse—muokkaa—vapauta-ratkaisumallia. Tässä mallissa käyttäjät kopioivat tietovarastosta itselleen työkopion. Jokainen voi työskennellä yksinäisesti ja muokata omaa työkopiota. Lopuksi ne yhdistetään uudeksi tiedostoksi. Versionhallintaohjelma usein auttaa yhdistämisessä, mutta lopuksi käyttäjä on vastuussa, miten se tapahtuu. Esimerkkinä Antti ja Ilkka kopioivat tiedoston A tietovarastosta. Molemmat tekevät muutoksia heidän tiedostoon A. Ilkka tallentaa ensin oman muutetun tiedoston, A1, tietovarastoon. Kun Antti yrittää tallentaa omaa muokattua tiedostoa, A2, tietovarastoon, niin tietovarasto ilmoittaa, että kyseinen tiedosto on muuttunut siitä, kun viimeksi Ilkka oli kopioinut sen. [3]



Kuva 4. Kopioi—muokkaa—yhdistä-ratkaisumalli osa 1. [3, muokattu]

Antin pitää yhdistää tietovarastossa oleva Ilkan tiedosto A1 omansa kanssa, jotta heidän tiedostonsa A1 ja A2 eivät ole ristiriidassa. Sen jälkeen Antti voi tallentaa tiedoston A3, jossa on yhdistettynä tiedostot A1 ja A2. Mikäli A1 ja A2 ovat ristiriidassa, niin Ilkka pystyy näkemään ristiriidan ja joutuu valitsemaan kumman muokkauksen ottaa ja yhdistää. Ratkaistuaan ristiriidan voi Antti turvallisesti tallentaa yhdistetyn tiedoston tietovarastoon. Yleensä versionhallintaohjelma auttaa ratkaisemaan ristiriitatilanteita, mutta lopuksi käyttäjä päättää, miten se tapahtuu, kuten yhditemisessäkin. [3]



Kuva 5. Kopioi—muokkaa—yhdistä-ratkaisumalli osa 2. [3, muokattu]

3.5 Muutosten havaitseminen

Yleisimpiä lähdekoodin muutoksien syitä on virheiden korjaaminen ja ominaisuuksien lisäys. Tehdyn muutoksen havaitseminen on hankalaa, koska muutokset voivat olla yksittäisissä tiedostoissa tai ne voivat olla monessa paikassa usean tiedoston sisällä. Jos tiedosto on pieni, niin yhdessä tiedostossa on helppoa verrata muutettua versiota alkuperäiseen versioon, mutta jos versiossa on paljon tiedostoja tai tiedosto on suuri, niin muutosta ei ole helppo löytää. Ongelman ratkaisemiseksi on keksitty erilaisia algoritmeja, joiden avulla muutokset voidaan havaita helpommin. Niiden avulla virheen etsiminen on helpompaa, koska ongelma yleensä löytyy viimeksi tehtyjen muutosten joukosta. [3]

3.6 Muutosten yhdistäminen

Yhdistäminen tarkoittaa toimintaa, jossa saman tiedoston kahdesta tai useammasta versiosta muodostetaan uusi versio. Kahden tiedoston yhdistämiseen on kolme erilaista tapaa. Yleensä usean tiedoston yhdistäminen tapahtuu yhdistämällä ensin kaksi tiedostoa ja niistä saatu tiedosto yhdistetään kolmanteen tiedostoon. Ensimmäinen

tapa on se, että jos molemmat versiot ovat sisällöltään samat, niin voidaan valita suoraan jompikumpi uudeksi versioksi. Toinen tapa on, että toinen aikaisempi versio on selvästi toista parempi. Voidaan ottaa parempi versio sellaisenaan uudeksi versioksi ja jättää huonompi kokonaan pois. Kolmas tapa ja yleisin tapa on se, että uusi versio kootaan kahdesta aikaisemman version parhaista osista ja niiden tulosta muokataan jonkin verran. Näiden lisäksi on mahdollista, että molemmat aiemmat versiot ovat huonoja, jolloin uusi versio aloitetaan tyhjästä. Tämä tiedosto ei kuitenkaan ole uusi versio vaan uusi tiedosto. [3]

Yhdistämismenetelmiä voidaan jakaa karkeasti kahteen luokkaan, tilapohjaiseen ja muutospohjaiseen. Tilapohjaiseen luokkaan kuuluvat ne menetelmät, jossa analysoidaan versioiden välisiä eroavaisuuksia, ja muutospohjaiseen luokkaan ne, jossa havaitaan tehtyjä muutoksia tutkimalla. Näiden ero on käytännössä se, että tilannepohjaisessa menetelmässä annettujen versioiden perusteella pitää selvittää tehdyt muutokset ja muutospohjaisessa tehdyistä muokkauksista on pidetty koko ajan lokia. Näiden lisäksi menetelmiä voidaan luokitella tietojen käsittely tasoihin, jotka ovat tekstipohjainen yhdistäminen, syntaksinen yhdistäminen, semanttinen yhdistäminen ja rakeenteellinen yhdistäminen. Tekstipojaisessa yhdistämisessä tietoa käsitellään puhtaana tekstinä, mikä tarkoittaa, että ristiriitakohdat ovat samaan kohtaan tehtyjä päällekkäisiä muutoksia; syntaksinen yhdistäminen, missä tietoa käsitellään ohjelmointikielen tasolla, jolloin menetelmä havaitsee ne ristiriidat, jotka aiheuttavat syntaksivirheitä; semanttisessa yhdistämisessä käsitellään ohjelmia niiden toiminnan perusteella, missä toiminnan muutokset havaitaan; rakenteellisessa yhdistämisessä ohjelma pitää päivittää. Ohjelmallisesti ei ole aina mahdollista havaita muutosten aiheuttamia ongelmia. [3]

3.7 Versionhallintajärjestelmät

3.7.1 SCCS

SCCS (Source Code Control System) oli ensimmäisiä varsinaisia versionkontrolli ohjelmia. Marc. J. Rochkindin kehitti SCCS:n vuonna 1972 SNOBOL4-kielelle, ja se myöhemmin muokattiin C-kielelle Unix-käyttöjärjestelmää varten. Vuosina 1975–1985 se oli ainoa järkevävaihtoehto versionhallinnassa. SCCS hallitsi UNIX-käyttöjärjestelmän

versionhallintaa, kunnes RCS kehitettiin. Ohjelma nykyäänkin toimii riittävän hyvin pienissä projekteissa, mutta sen käyttö niissä ei ole suositeltavaa. [4]

3.7.2 RCS

RCS (Reversion Control System) on 1980 luvulla Walter F. Tichyn kehittämä yhden käyttäjän vapaaversiohallinta järjestelmä, koska siinä ei ollut minkäänlaisia verkko-ominaisuuksia. Ohjelman alkuperäisenä tarkoituksena oli olla ohjelmistokehittäjän työkalu. Ohjelma pystyi palaamaan aiemmin toimineeseen versioon virheen sattuessa niin, järjestelmien ylläpitäjät alkoivat tallentaa muokattuja tiedostoja, mistä syntyi käyttötapa ylläpitäjille. Ohjelman yksi tärkeimmistä parannuksista on vanhojen versioiden käsittelynopeus. RCS pystyi versioimaan yksittäisiä tiedostoja, mutta tiedostot voivat olla mitä tahansa dokumentteja. RCS käyttää lukitse—muokkaa—vapauta ratkaisumallia. [4]

3.7.3 CVS

CVS (Concurrent Version System) on Dick Grunen ja hänen kahden oppilaansa RSC:n pohjalta kehittämä vapaa, monen käyttäjän versionhallintajärjestelmä, mikä mahdollistaa usean kehittäjän samanaikaisen muokkaamisen. Kopioi—muokkaa—yhdistä-ratkaisumallin ominaisuuksien avulla mahdollistettiin usean kehittäjän samanaikainen toiminta. Linuxin suosion ja Internetin yleistymisen takia ohjelman suosio alkoi kasvaa, ja lopulta se sai de facto -standardin aseman. CVS pitää kirjaa tiedostojen sijaintihakemistosta ja osaa yhdistää useita hakemistoja yhdeksi kokonaisuudeksi sekä mahdollistaa versioinnin, eli reversiopuuhun luotua haara voi myöhemmin yhdistää päähaaraan. Reversion ja version välisen eron tekee se, että reversiot ovat versioita yksittäisestä tiedostosta ja ne numeroidaan samaan tapaan kuin RCS:ssä. [4]

3.7.4 GIT

Linus Torvalds kehitti C-kielellä Linux-ytimen kehittämistä varten oman hajautetun versionhallinnan. Git-ohjelman ajatus on olla mahdollisimman nopea ja tukea hajautettua ohjelmistokehitystä, joka toimisi hyvin, vaikka tekisi töitä ison projektin ja

pitkän historian kanssa. Ohjelma koostuu useista pienistä työkaluohjelmista. Lisäksi mukana on joukko skriptejä, joista saadaan miellyttävä käyttöliittymä alemman tason työkaluihin. Ohjelma käsittelee muutoksen koodin kokonaisuutena eikä yksittäisen tiedoston pohjalta. [4]

3.7.5 Subversion

Subversion (SVN) tehtiin CVS:n pohjalta, mutta tarkoituksena oli vain korjata CVS:ssä havaittuja puutteita ja lisätä tarpeellisiksi koettuja ominaisuuksia. Se pyrittiin pitämään samanlaisena kuin CVS, jotta niiden käyttäjien oli helppo vaihtaa SVN-ohjelmaan. Lisäominaisuutena CVS:ään tuli commit-toiminto, mikä tekee sen, että tiedostoa siirtäessä tietovarastoon se joko menee kokonaan tai se ei mene ollenkaan. Attribuuttien tai tiedostojen nimien muuttaminen toimii samalla tavalla kuin sisällön muokkaaminen. Se hallinnoi tiedostojen lisäksi myös kokonaisten hakemistojen historiaa. [4]

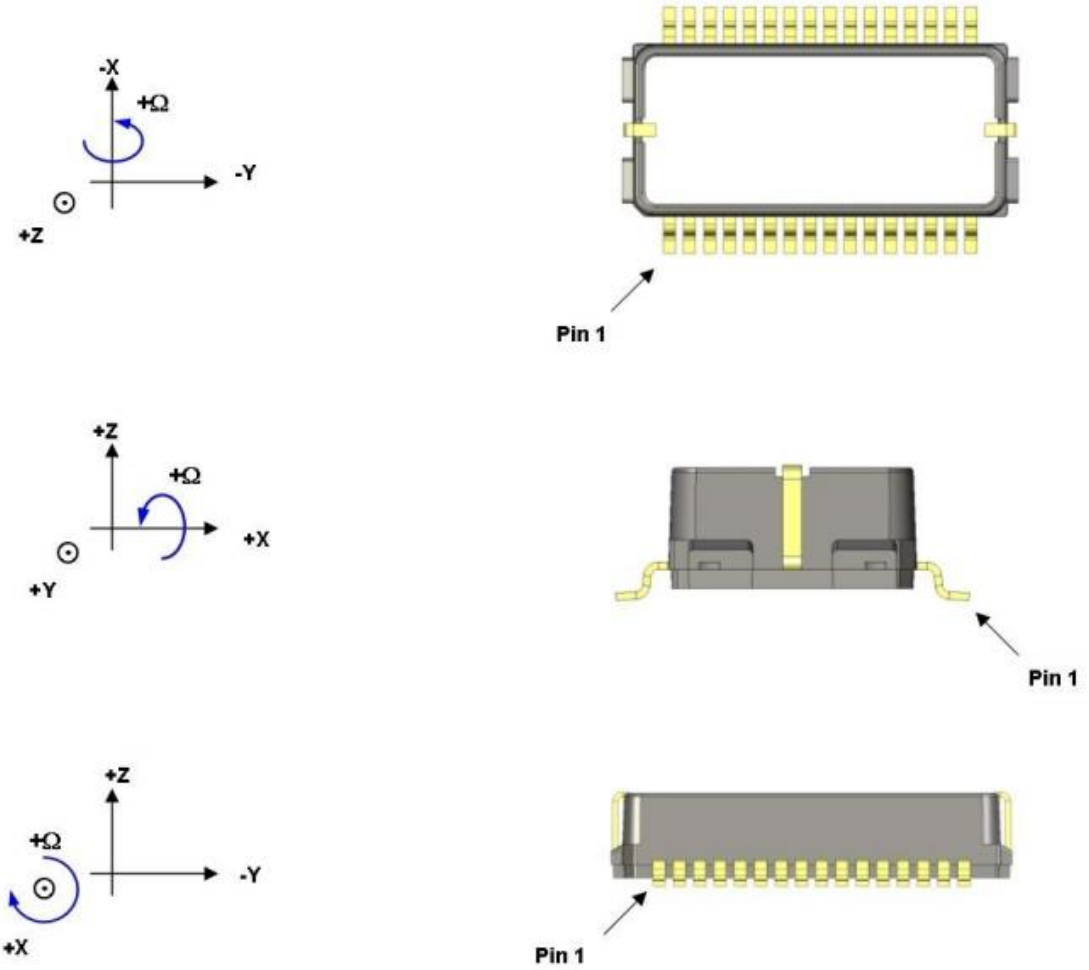
4 Tortoise SVN

Tortoise SVN on ilmainen Windows-käyttöliittymään oleva avoin lähdekoodi, Subversion -versionhallintaohjelma. Ohjelma on integroitunut täysin Windows-resurssinhallintaan. Versioitujen tiedostojen tai kansioiden tila ilmaistaan pienillä kuvakepäällysteillä, joiden avulla voi nopeasti nähdä työkopion tilan. Ohjelmassa on graafinen käyttöliittymä. Kun luettelee muutokset tiedostoon, voi klikata versiota nähdäkseen siihen liittyvät kommentit. Koska Tortoise SVN pohjautuu Subversioon, sillä on etuna muun muassa versioidut hakemistot, missä SVN toteuttaa versioidun tietojärjestelmän, joka tallettaa koko kansiopuun muutokset. Subversion-järjestelmässä tietovarastoon kytkeytyminen on irrotettu omaksi rajapinnaksi, mikä helpottaa uusien kytkeytymistapojen kehittämistä. Ohjelma esittää tiedostojen eroavuudet binäärisenä erottelualgoritmina, joka toimii identtisesti teksti- ja binääri-dataa sisältävissä tiedostoissa. [3]

5 Muratan yhdistelmäanturi

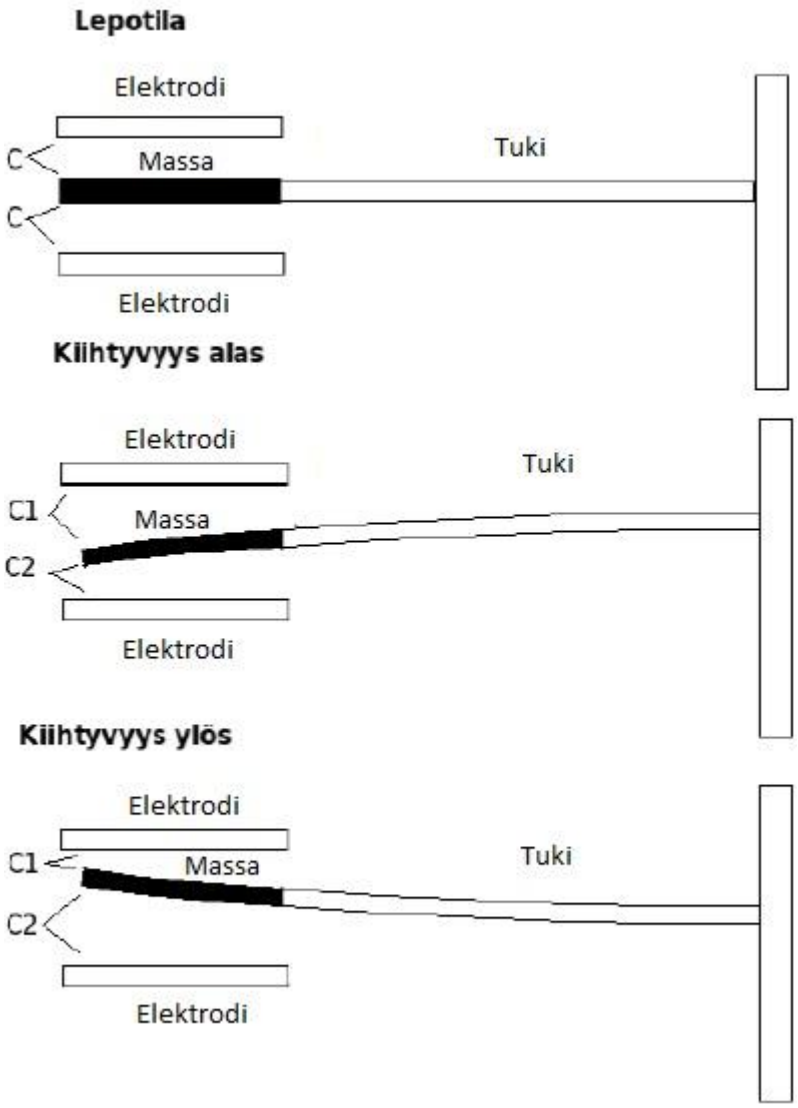
Murata Electronicsin päätuotteina ovat 3D MEMS -teknologiaan perustuvat kiihtyvyys-, kallistus-, kulmanopeus- ja yhdistelmäanturit. Tämä työ perustuu yhditelmäantureihin.

Yhdistelmäantureita on 1- tai 3-akselisina ja eri alueella. Kuvassa 6 näkyvät yhdistelmäanturin akselit ja kiertosuunnat. [6]



Kuva 6. Yhdistelmäanturin akselit ja kiertosuunta. [6]

Kiihtyvyyden mittaus perustuu massojen liikkeiden havainnointiin. MEMS-anturit perustuvat kapasitiiviseen mittaustekniikkaan. [5] Anturi mittaa liikettä ja liiketilan muutoksia. Kapasitiivisessä MEMS-anturissa on taipuva tuki, jonka päässä on massa. Massa toimii liikkuvana elektrodina ja sen kummallekin puolella on staattinen elektrodi. Kiihtyvyyden muutos lähentää massaa kohti toista elektrodia, jolloin tuen ja lähemmän elektrodin kapasitanssi kasvaa. Vastaavasti etäisyys toiseen elektrodiin kasvaa ja kapasitanssi pienenee. Kuvassa on kiihtyvyysanturin toimintaperiaate. Anturi muuttaa rakenteen muutoksen sähkövirraksi. [5;7]

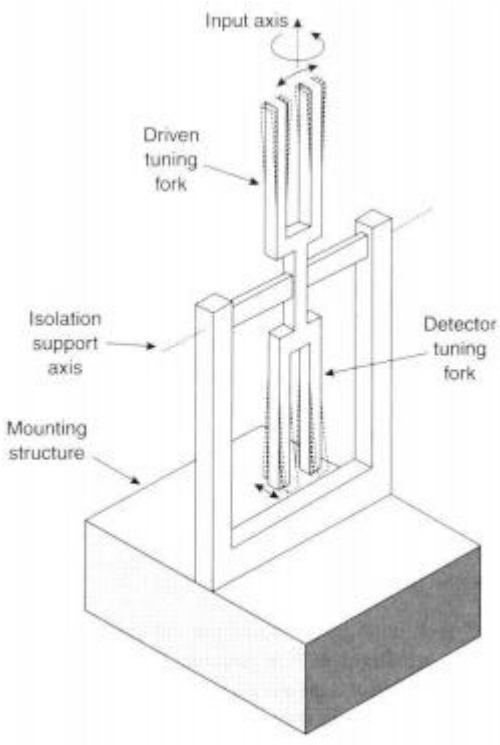


Kuva 7. Kiihtyvyysanturin toimintaperiaate [7,muokattu]

Kapasitiivinen anturielementti mittaa kiihtyvyyttä sekä positiiviseen että negatiiviseen suuntaan ja on herkkä staattiselle kiihtyvyydelle ja tärinälle. Kiihtyvyyden tunnistavat elementit on tehty yksikiteisestä piistä ja lasista. Se tekee anturista aikaan ja lämpötilaan nähden poikkeuksellisen luotettavan ja takaa ennennäkemättömän tarkkuuden ja erinomaisen vakauden. [7]

Kulmanopeusanturissa on haarukka, joka laitetaan resonoimaan sähkövirran avulla. Anturin pyöriessä sisäänmenoakselin ympäri Coriolis-voima aiheuttaa edestakaisen liikkeen, joka on kohtisuorassa pakotetun resonoinnin suuntaan ja sisäänmenoakselin suuntaan nähden (nuolet alemmassa haarukassa). Tätä liikettä mitataan

kapasitiivisesti, ja tuloksena on kulmanopeudella moduloitu signaali, josta saadaan kulmanopeus selville. [11]



Kuva 8. Kulmanopeusanturin toimintaperiaate [11]

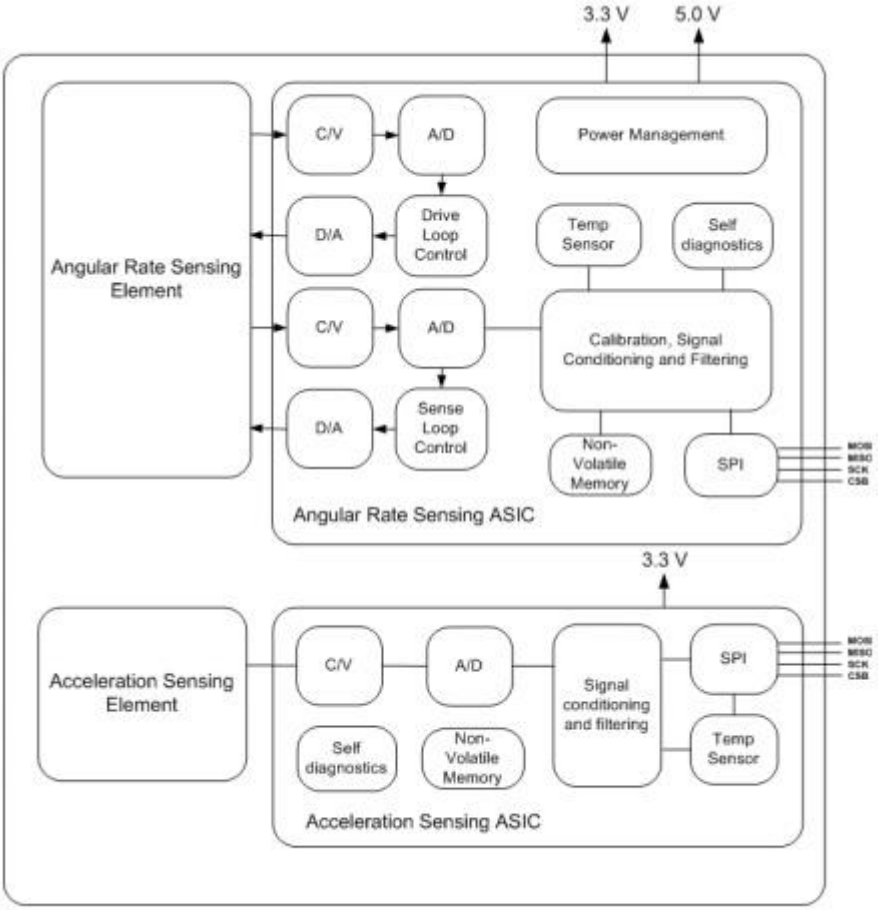
5.1 Käyttökohteet

Murata Electronicsin yhdistelmäanturit menevät pääasiassa autoteollisuuteen. Lähes kaikki automerkit käyttävät MFI:n antureita. Yhdessä autossa voi olla useita antureita. Vaikka antureita löytyy 3-akselin suuntaisesti mittaavia, käyttävät monet auton turvajärjestelmät omia antureita. Antureita käytetään esimerkiksi lukkiutumattomissa jarruissa (ABS), ajonvakautuksessa (ESC), mäkilähtöavustimissa (HSA), sähköisessä käsijarrussa (EPB), kaatumisen estossa (ROV), aktiivisessa jousituksessa (ECS) ja auton kallistumisen seuraamisessa (ARC). [5]

5.2 Yleiset ominaisuudet

Kiihtyvyysanturit mittaavat kiihtyvyyden muutokset 3D MEMS -teknologialla valmistetulla anturielementillä. Yhdessä anturielementissä voi olla kolmeen eri suuntaan reagoivaa massaelementtiä. ASIC-piiri muuntaa anturielementin signaalin

digitaaliseksi ulostuloksi. Kuvassa 9 on esitettynä yhditelmäanturin piirikaavio. Kuvassa vasemmalla alhaalla on kiihtyvyysanturielementti (3D MEMS) ja tämän oikealla puolella kiihtyvyysanturin ASICin piirikaavio. Vasemmalla ylhäällä on kulmanopeudenanturielementti ja tämän oikealla puolella kulmanopeudenanturin ASICin piirikaavio. [6]



Kuva 9. Yhdistelmäanturin piirikaavio. [6]

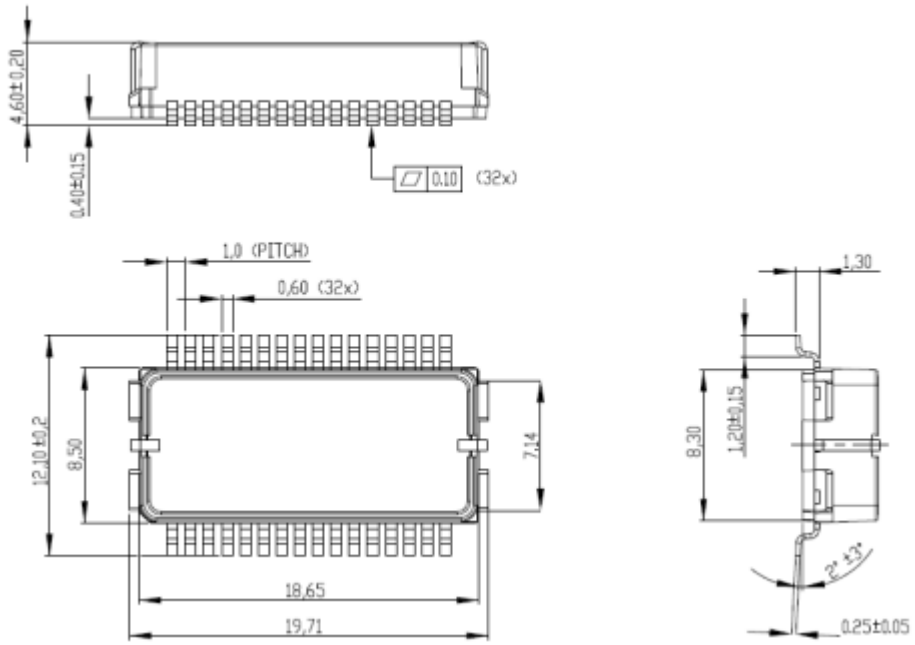
5.3 3D MEMS -anturi

Muratan termi 3D MEMS tarkoittaa piikiekkoon kuivasyövytyksellä valmistettuja kolmiulotteisia mekaanisia rakenteita ja niiden eristämistä ulkomaailmasta lasi-piiteknologiaan perustuvilla kansikiekoilla. Yhdistelmä on ainutlaatuinen ja perustuu Muratassa kehitettyihin teknologioihin. Piiantureissa käytetään yksikiteistä piitä ja lasia. Rakennekiekolle syövytetään jousimassarakenteita, mittauskondensaattoreita ja sähköstaattisen voiman synnyttäviä toimielimiä. Kansikiekon lasialueilla eristetään läpivientirakenteet piikiekosta. Rakenne- ja kansikiekko yhdistetään anodisella toisiinsa,

jolloin liikkuvat rakenteet saadaan suojattua lialta ja kosteudelta. MEMS-osan sisällä voi olla lähes täydellinen tyhjiö. MEMS-osan toimintaan voidaan vaikuttaa kaasuvaimennuksella. Kaasuvaimennuksella voidaan vaimentaa massaelementin mekaanista värähtelyä sekä vaimentaa ulostulon sähköistä kohinaa. [5]

5.4 Kotelointi

Yhdistelmäantureissa käytetään vain yhtä koteloa. Kotelossa on 32 kontaktijalkaa. Kotelo on tyypiltään Dual In-Line package (DIP), joka tarkoittaa, että kotelo on mikä tahansa suorakulmainen pakkaus, jossa on kaksi tasaisen välimatkan päässä toisistaan rinnakkaisen tappiriviä osoittaa alaspäin [8]. Kuvassa 4 on esitettynä kotelon mitat millimetreinä. Kotelon kontaktointijalat ovat 0,60 mm leveät ja kotelon pohjan tasolla. Jalkojen sijainti parantaa jalkojen mekaanista kestävyyttä. [6]



Kuva 10. Yhdistelmäanturin spesifikaatiot. [6]

5.5 Kokoonpano

Yhdistelmäantureiden kokoonpanovaiheessa anturi- sekä ASIC-elementit asennetaan koteloon. Elementit kiinnitetään kotelon pohjaan erikoisliimalla. Elementtien, ASICien ja kotelon välinen sähköinen kontaktointi tehdään kultalangalla. Lopuksi kotelo valetaan

täyteen elastista suojageeliä. Suojageelin vesipitoisuus on hyvin pieni ja vesihöyryn läpäisevyys erittäin vähäinen. Lopuksi päälle kuumamuovataan metallinen suojakansi.

5.6 Testaus ja kalibrointi

Kokoonpanon jälkeen kaikki tuotteet testataan ja kalibroidaan. Ensimmäisenä testauksessa tuotteita vanhennetaan lämpö- ja kylmätestilaitteissa. Kalibrointi tehdään kääntelemällä tuotteita ja mittaamalla maanvetovoiman aiheuttamaa kiihtyvyyttä. Samat mittaukset ja kalibroinnit täytyy tehdä myös kylmä- ja kuumatestereillä, koska tuotteissa on lämpötilakompensointi. Ennen asiakkaalle pakkaamista tuotteet vielä erotellaan johdinkehyksiltä.

6 ESEC 3100 PLUS wire bonder -laite

ESEC 3100 plus wire bonder -laite käyttää Esecin TY-teknologiaa. Kinematiikan avulla laite pystyy toimimaan nopeasti ja vähentää tärinää käyttäen Oerlikon kokoonpano-osia. Perinteisissä wire bonder -laitteessa bondauspää on asennettu kohtisuorassa x/y-tasoon. Esec 3100 plus wire bonder -laitteessa TY-teknologia on linkitetty pyörimiseen lineaarisen liikkeeseen luodakseen theta/y-liikkeen. Pyörimisen käyttö mahdollistaa massan keskittämistä lähelle theta-akselia, mikä vähentää massan hetkellistä momenttia. Pienen momentin ja kääntyvän pään ansiosta saavutetaan suurempi kiihtyvyys pienemmällä moottorin voimalla. Bondheadia kontrolloidaan tarkalla ilmansyöttö-teknologialla. Laitteessa on ohjelmoitava lämmittimen korkeuden säätö, joka takaa kapilaarin iskun olevan kohtisuorassa bondauksen nollakohtaa. Siinä on myös ainutlaatuinen ilmansyöttöteknologia, mikä vaikuttaa laitteen tarkkuuteen. [9]

Kuva 11. Esec 3100 wire bonder laite

7 Suunnittelu

Projektin tehtävänä oli selvittää, missä laitteissa Tortoise SVN -versionhallintaohjelma toimii, saattaa toimia pienillä muutoksilla ja missä ei toimi. Tortoise SVN pohjautuu Windows-käyttöjärjestelmään ja vaatii, että laite olisi verkossa. Koska ohjelmaa on todella hankala seurata laitteella, lähdin selvittämään, mitä käyttöjärjestelmää kukin laite käyttää ja onko kyseinen laite verkossa. Tutkittua laitteita tehtiin taulukko 1, jonka avulla tehtiin taulukko 2 perustuen tietolähteisiin ja osaamiseeni.

Taulukko 1. Laitteiden käyttöjärjestelmät ja verkkotila

Laitteet	Käyttöjärjestelmä	Verkossa
Datacon die bonder (3 kpl)	Linux	kyllä
Esec die bonder (2 kpl)	Windows XP empedded SP3	kyllä
Esec 3100 series wire bonder (3 kpl)	Windows 2000 SP4	kyllä
Iconn wire bonder (2 kpl)	Windows XP 2002 SP3	kyllä
Esec 2007 series die bonder 09 (1 kpl)	MS Dos	ei
Esec series wire bonder (1kpl)	MS Dos	ei

Murata Electronicsin kokoonpanon Cobra-linjan laitteet ovat:

- Yksi datacon die bonder -laite liimaa kallistusanturin elementit koteloon.
- Kaksi Datacon die bonder -laitetta ja kaksi Esec die bonder -laitetta, jotka liimaa kulmanopeusantureiden asicit ja elementit koteloon.
- Kolme Esec wire bonder -laitetta ja kaksi Iconnin wire bonder -laitetta, jotka tekevät kultalangalla sähköisen kontaktin kulmanopeusantureiden, asicien ja kotelon välille.
- Yksi Esec die bonder -laite, joka liimaa kallistusanturin asicit koteloon.
- Yksi Esec wire bonder -laite, joka tekee kultalangalla sähköisen kontaktin kulmanopeusantureiden, asicien ja kotelon välille.

Taulukko 2. Mitkä koneet saadaan toimimaan

Laitteet	Toimii	Saattaa toimia	Ei saa toimimaan
Datacon die bonder (3 kpl)		X	
Esec die bonder (2 kpl)	X		
Esec wire bonder (3 kpl)	X		
Iconn wire bonder (2 kpl)		X	
Die bonder 09 (1 kpl)			X
Wire bonder 06 (1kpl)			X

8 Toteutus

Lähdettiin kokeilemaan Tortoise SVN nimisen versionhallintaohjelman käyttöä toimiston tietokoneella. Asentaessaan ohjelmaa huomattiin, että uusimman Tortoise SVN:n päivitys ei sovellu vanhoihin Windows-käyttöjärjestelmiin joten ladattiin Tortoise SVN 1.7.5:n joka tukee Esec wire bonder -laitteen Windows 2000 SP4 -käyttöjärjestelmää. Kokeiltaessa ohjelmaa esiintyi ongelmia. Luotaessa ja päivitettäessä arkistoa ohjelma hakee suoraan Windowsin käyttöjärjestelmästä käyttäjän eikä anna muuttaa sitä. Ohjelman tarkoitus on olla tuotantolinjalla, missä lähes kaikilla koneilla käyttäjätunnus on sama. Täten ohjelma ei auttaisi alkuperäistä tavoitetta. Manuaalista löydettiin ohjeet, että halutessa muuttaa tekijäntietoja tai viestiä lokista pitää määrittää pre-revprop-change-komentosarja. Internetin kautta löytyi mallikomentosarjoja, mutta malleissa ei ollut komentosarjaa, jossa voidaan muuttaa tekijätietoja sekä lokiviestiä. Etsittiin Internetistä yksi komentosarja malli (liite 1), jossa oli mahdollisuus pelkästään muuttaa lokiviestiä, ja muokkasin sen sellaiseksi, missä voidaan muuttaa tekijätietoja ja lokiviestiä (liite 2).

Versionhallintaohjelman hyviä ominaisuuksia on vertailu, jossa pystyy vertailemaan muutetun tiedoston sisältöä halutun tiedoston kanssa. Ladattuaan tuotannon wire bonder -laitteelta reseptit toimiston tietokoneelle huomattiin, että reseptit olivat WBRCP-tiedostomuotoa. Kokeiltiin avata kyseinen tiedostomuoto Notepadillä, mutta tiedostosta ei saanut mitään selvää. Otettiin yhteys ESEC:n asiantuntijaan ja hän vastasi, että tiedostoja ei saa muutettua selväksi koodikieleksi. Ainoa tapa saada ohjelmanvertailuominaisuus ESEC 3100 wire bonder -laitteella on laitteen omalla ohjelmalla luoda txt- tai xml-tiedosto reseptistä, josta näkyy kaikki speksit, mutta kyseisen tiedoston luontia ei saa mitenkään automaattiseksi. Se merkitsee, että prosessituen henkilöllä on lisää työtä saada se tehtyä.

Ongelmien ratkottua lähdettiin asentamaan ohjelmaa harjoittelulaitteelle. Tavoitte oli tehdä yhdestä toimistoverkon verkkolevyistä arkisto ja laitteisiin työkopiot, joita voidaan muokata laitteen mukaan. Murata Electronics käyttää Novell-verkkokäyttöjärjestelmää. Esec 3100 wire bonder -laite sekoaa, kun siihen asennetaan kyseinen ohjelma, joten tuotantoverkosta tiedostojen siirto toimistoverkkoon toi lisää ongelmia. Ensimmäiseksi kokeiltiin FTP:n kautta ottaa yhteyden toimistoverkkoon,

mutta Tortoise SVN -ohjelma ei saanut asetettua FTP:tä arkistoksi. Toiseksi yritettiin erilaisia FTP driver -ohjelmia eli ohjelmia, jotka näyttäisivät FTP:n kiinteänä asemana. Siinä Tortoise SVN sai tehtyä arkiston FTP-asemalle, mutta wire bonder -laitteen oma bondausohjelma ei löytänyt kyseistä asemaa. Kolmanneksi kokeiltiin Windowsin omaa kansion jakamista verkossa. Sen avulla saatiin Tortoise SVN:n toimimaan, ja kyseinen kansio näkyi wire bonder -laitteen omassa ohjelmassa, mutta kansiosta lataaminen ja tallentaminen eivät aina onnistuneet.

9 Yhteenveto

Alkutilanne oli saada Muratan komponenttikokoonpanossa laitteiden reseptien muutokset ja niiden syiden tiedonkulku paremmaksi eri vuorojen välillä. Työ jakautui kolmeen vaiheeseen:

- tutkia eri versionhallintaohjelmia ja niiden soveltuvuutta kokoonpanolinjalle
- testata ohjelmaa ja muokata sitä kokoonpanolle sopivaksi
- ottaa käyttöön.

Ensimmäisessä vaiheessa tutkittiin monta eri ohjelmaa, mutta päätettiin lähteä kokeilemaan Tortoise SVN -ohjelmaa, mikä on jo yrityksessä käytössä, koska sen ohjelman toimivuudesta on jonkin verran osaamista komponenttivalmistuksen testauspuolella. Lisäksi ohjelman pitäisi soveltua joihinkin komponenttivalmistuksen laitteiden käyttöjärjestelmiin. Toisessa vaiheessa huomattiin, että ohjelma tarvitsi pieniä muutoksia, jotta se sopisi kokoonpanolinjalle. Ratkaistiin ongelma lisäämällä koodin pätkä ohjelman asetuksiin. Kolmannessa vaiheessa tuli yrityksen verkkojen kanssa hankaluuksia. Yritettiin saada toimistoverkon asemasta arkiston, jotta toimistossa olevalla koneella oleva henkilö pystyisi näkemään muutokset ja selitykset. Aikaa meni verkkojen välisten yhteyksien saamisessa ja IT-osaston kommunikoinnin kanssa niin paljon, ettei saatu ratkottua verkkojen välistä yhteyttä, joka toimisi Tortoise SVN:llä ja ESEC wire bonder -laitteella.

Suositukseni yritykselle oli, että versionhallintaohjelma on hyvä ratkaisu tiedostojen muutosten ja muutosten syiden tiedonkulkuun, mutta ohjelma ei sovellu kovin hyvin Muratan komponenttikokoonpanolinjalle, koska ei löytynyt yhtään ohjelmaa, joka tukisi

sekä Ubuntua ja Windowsia. Lisäksi reseptien muutoksien parametrit eivät näy kaikissa resepteissä. Ehdotukseni Murata Electronicsille on, että jos yritys aikoo ottaa versionhallintaohjelman käyttöön kokoonpanossa, se vaatisi henkilön, joka osaa verkkojen väliset yhteyksien toiminnat hyvin. Toinen mahdollisuus on hankkia tuotantoverkkoon verkkoasema, joka ei ole Novell -verkkokäyttöjärjestelmässä.

Lähteet

1 versionhallinta: linux.ictlab.kyamk.fi/esitykset/2006_2/versionhallinta.odp. Luettu 12.4.2013

2 versioncontrol: http://svnbook.red-bean.com/en/1.7/svn-book.pdf. Luettu 5.6.2012

3 Tortoise SVN: http://tortoisesvn.net/docs/release/TortoiseSVN_en/index.html. Luettu 5.6.2012

4 versioncontrol: http://code.google.com/p/pysync/wiki/VCSHistory. Luettu 5.6.2012

5 gyroscope: http://www.muratamems.fi/en/about-murata/murata-electronics-oy/mems-technology. Luettu 12.4.2013

6 gyroscope: http://www.muratamems.fi/sites/default/files/documents/82113100d_scc1300-d04_datasheet1.pdf. Luettu 12.4.2013

7 kiihtyvyysanturi: http://lib.tkk.fi/Dipl/2007/urn009901.pdf. Luettu 12.4.2013

8 dual in line package: https://www.princeton.edu/~achaney/.../Dual_in-line_package.html. Luettu 12.4.2013

9 Esec 3100 wire bonder: http://www.oerlikon.com/ecomaXL/get_blob.php?name=Brochure_WB3100Plus.pdf&download=1. Luettu 12.4.2013

10 Murata Electronics: http://www.muratamems.fi/en/about-murata/murata-electronics-oy. Luettu 20.8.2012

11 kulmanopeusanturi: http://www.tkt.cs.tut.fi/kurssit/2540/paikannuspruju.pdf. Luettu 12.4.2013

Internetistä haettu komentosarja, jossa voidaan muuttaa vain viestintälokia.

```
@ECHO OFF
:: Set all parameters. Even though most are not used, in case you want to add
:: changes that allow, for example, editing of the author or addition of log messages.
set repository=%1
set revision=%2
set userName=%3
set propertyName=%4
set action=%5

:: Only allow the log message to be changed, but not author, etc.
if /I not "%propertyName%" == "svn:log" goto ERROR_PROPNAME

:: Only allow modification of a log message, not addition or deletion.
if /I not "%action%" == "M" goto ERROR_ACTION

:: Make sure that the new svn:log message is not empty.
set bIsEmpty=true
for /f "tokens=*" %%g in ('find /V ""') do (
set bIsEmpty=false
)
if "%bIsEmpty%" == "true" goto ERROR_EMPTY

goto :eof

:ERROR_EMPTY
echo Empty svn:log messages are not allowed. >&2
goto ERROR_EXIT

:ERROR_PROPNAME
echo Only changes to svn:log messages are allowed. >&2
```

```
goto ERROR_EXIT

:ERROR_ACTION
echo Only modifications to svn:log revision properties are allowed. >&2
goto ERROR_EXIT

:ERROR_EXIT
exit /b 1
```

Muuneltu komentosarja, jossa voidaan muuttaa käyttäjän tunnuksia ja viestintä lokia.

```
@ECHO OFF
:: Set all parameters. Even though most are not used, in case you want to add
:: changes that allow, for example, editing of the author or addition of log messages.
set repository=%1
set revision=%2
set userName=%3
set propertyName=%4
set action=%5

:: Allow to modify author.
REM pre-revprop-change.bat hook script
if "%4" == "svn:log" exit /b 0
if "%4" == "svn:author" exit /b 0
echo "Changing revision properties %4% is prohibited" >&2



:: Only allow modification of a log message, not addition or deletion.
if /I not "%action%" == "M" goto ERROR_ACTION

:: Make sure that the new svn:log message is not empty.
set bIsEmpty=true
for /f "tokens=*" %%g in ('find /V """') do (
set bIsEmpty=false
)
if "%bIsEmpty%" == "true" goto ERROR_EMPTY

goto :eof

:ERROR_EMPTY
echo Empty svn:log messages are not allowed. >&2
```

```
goto ERROR_EXIT

:ERROR_PROPNAME
echo Only changes to svn:log messages are allowed. >&2
goto ERROR_EXIT
```

Yhdistetty komentosarja, jossa voidaan muuttaa viestintälokia ja käyttäjän tunnuksia.

```
@ECHO OFF
:: Set all parameters. Even though most are not used, in case you want to add
:: changes that allow, for example, editing of the author or addition of log messages.
set repository=%1
set revision=%2
set userName=%3
set propertyName=%4
set action=%5

:: Allow to modify author.
REM pre-revprop-change.bat hook script
if "%4" == "svn:log" exit /b 0
if "%4" == "svn:author" exit /b 0
echo "Changing revision properties %4% is prohibited" >&2



:: Only allow modification of a log message, not addition or deletion.
if /I not "%action%" == "M" goto ERROR_ACTION

:: Make sure that the new svn:log message is not empty.
set bIsEmpty=true
for /f "tokens=*" %%g in ('find /V """') do (
set bIsEmpty=false
)
if "%bIsEmpty%" == "true" goto ERROR_EMPTY

goto :eof

:ERROR_EMPTY
```

```
echo Empty svn:log messages are not allowed. >&2
goto ERROR_EXIT

:ERROR_PROPNAME
echo Only changes to svn:log messages are allowed. >&2
goto ERROR_EXIT

:ERROR_ACTION
echo Only modifications to svn:log revision properties are allowed. >&2
goto ERROR_EXIT

:ERROR_EXIT
```